NAVAL POSTGRADUATE SCHOOL
Monterey, California

THESIS

GAME PLAYING -- THE ANCIENT GAME OF GO

by

Glenn Fredric Gottschalk

Thesis Advisor: Gregory D. Gibbons

December 1971

Approved for public release; distribution unlimited.

KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Game Playing						

Game Playing -- The Ancient Game of GO

by

Glenn Fredric Gottschalk
Lieutenant, United States Navy
B.S., United States Naval Academy, 1968

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN COMPUTER SCIENCE

from the

NAVAL POSTGRADUATE SCHOOL
December 1971

Author Glenn Fredric Gottschalk

Approved by: Gregory Dean Richards
Thesis Advisor

R. E. Garkell
Chairman, Department of Mathematics

Academic Dean

ABSTRACT

This paper contains a description of a game playing program which plays the game of GO. The paper discusses the strategy and approach used by the program. The primary objective of the project was to develop a game playing program that would generate moves which would be considered intelligent if made by a human being. A brief description of previous approaches to game playing is given with a comparison between these approaches and those used by the program.

TABLE OF CONTENTS

I. INTRODUCTION

Artificial Intelligence is the field of Computer Science which concerns itself with the demonstration of intelligent behavior by computers. Artificial Intelligence is a fairly new field of research which is extremely complex and requires extensive research to investigate all the areas it encompasses. The field is too new to have allowed extensive research to be undertaken in all areas. A considerable amount of research has been done in some areas, with several conclusions being made, while other areas have received little or no research. The complexity and depth of the field has led to the development of several approaches to research. None of the approaches has of yet proven to be dominant.

A. DEVELOPMENT

The question "Can a computer think?" plays an important part in the development of Artificial Intelligence. Much discussion has been generated about the question since the inception of Artificial Intelligence. The essence of the discussions has been that the answer to the question depends on the definition of "think." To avoid the conflict of definition, Artificial Intelligence research is based on the proposition that computers can perform functions which would be considered intelligent behavior if done by a human being. Researchers in the field of Artificial Intelligence do not generally consider or argue the definition of "think."

The proposition that computers can act intelligently is often disputed on the basis of the statement "Computers can only do what they are told, they are simply tools of man." This statement is undeniably true but does not, however, prove that computers can not demonstrate behavior considered intelligent in humans. This misconception is nicely discussed by Minsky (1968, p. 11):

> One of the most popular misconceptions about Artificial Intelligence is that problem solving by computers is confined to precisely defined "formal" problems. This is based in part on the dreadfully misleading set of concepts that people get when they are told (with the best intentions) that computers are nothing but assemblies of flip-flops; that their programs are really nothing but sequences of operations upon the binary numbers, and so on. While this is one useful viewpoint, it is equally correct to say that the computer is nothing but an assembly of symbol-association and process-controlling elements and that programs are nothing but networks of interlocking goal-formulating processes. This latter attitude is actually much healthier because it reduces one's egotistical tendency to assume total comprehension of all possible future implications.

The goal of Artificial Intelligence, therefore, is to develop processes which produce results that would be considered intelligent if produced by humans. These processes could then be implemented in the computer by a program.

An objective of Computer Science in general and Artificial Intelligence in particular is to make humans and machines partners working together to solve complex problems. Each would perform those tasks or functions which it does best. Interaction between the two would be essential if a partnership was to be maintained. An objective of Artificial Intelligence is to develop means by which the computer could take

a larger part of the problem-solving duties away from man so that he can spend more time on creativity. This project was to develop and test problem-solving techniques that might be beneficial in obtaining this objective.

B. APPROACHES TO ARTIFICIAL INTELLIGENCE

The approaches to Artificial Intelligence can be divided into two general categories. One of the categories is the attempt to simulate human behavior in problem-solving tasks. That is, to duplicate human problem-solving methods by a computer program. The other category consists of producing good results by any method possible. The second category does not emphasize imitating human problem-solving methods.

The main promoters of the first category are Newell, Simon, and Shaw. Work in this area -- Simulation of Human Thought -- has focused rather sharply at Carnegie Institute of Technology, where the group is led by Newell and Simon. Their basic assumption is that it is advantageous to know and understand how to program a computer to perform the same problem-solving tasks as humans. They are as concerned with the overall schematic strategy as they are with the produced results.

The principal promoters of the second category are located at Massachusetts Institute of Technology and Stanford University. The philosophy of this group is that the effective path to progress in Artificial Intelligence lies in producing good results by any means possible. While they do not disregard the human simulation approach category, they do not limit the research to this area. They believe that to restrict

the computer to human simulation is not using the computer's total capability. This conclusion could be based on the premise that since the computer does not physically resemble man, it should not be required to duplicate his problem-solving processes.

C. GAME PLAYING

Game playing is one area of research in Artificial Intelligence. It provides the researcher with many fascinating problems. In general, game playing provides a useful environment in which to study the nature and structure of complex problem-solving processes.

The objective of game playing is to use the game playing environment as a means for testing processes, rather than to produce the world's best player at a particular game. The game playing environment serves as the experimental apparatus for the researcher. Data collected from such an experimental apparatus can be used to derive various conclusions as to the feasibility of specific problem-solving processes not necessarily limited to game playing.

Both categories of approaches to Artificial Intelligence are applicable to game playing. Newell, Simon, and Shaw used the game of CHESS to simulate human problem-solving techniques. The program contained adaptive and selective search techniques. These search techniques are processes that analyze game situations and extract data. The game of CHESS was chosen because its environment is to complex that not all possible moves and situations can be explored prior to

making a move. To explore all possible moves and board situations is known as the brute-force method. This method cannot be used in CHESS because the number of possible move paths is approximately 10^{120} (Feigenbaum, p. 5). Assuming that one move path can be investigated in a microsecond, the search would still take approximately $3X10^{109}$ centuries. To limit the search time, a certain amount of analysis and generalization must be performed in order to select a move. In behavioral science, analysis and generalization are considered to be characteristics of intelligence. Therefore, if the computer can be programmed to do analysis and generalization, the computer would be demonstrating some characteristics of intelligence.

Professor A. L. Samuel of Stanford University used the game of CHECKERS to produce a game playing program that would be classified in category two. Like CHESS, CHECKERS cannot be handled by the brute-force method. Samuel's objective was to produce a CHECKER-player that would play a good game of CHECKERS. His stated goal was to produce the CHECKER-player by any means possible. Samuel judged his success on the playing ability of his program, not whether he had duplicated the human problem-solving ability. Samuel's program played an excellent game of CHECKERS. In addition, Samuel's program was capable of some learning. The learning capability was divided into two areas. One area was known as rote learning and consisted of saving all the board situations encountered during play, together with their computed value in relation to the advantage or disadvantage of each.

The second area, the most significant, is concerned with changing the weighted value of parameters during the play of the game. The parameters are used in a polynomial to determine the goal to be achieved by a particular move.

Although both categories of approaches have demonstrated certain advantages and disadvantages, neither approach clearly dominates the other. Samuel's CHECKER-player played a superior game in comparison to Newell, Simon, and Shaw's CHESS-player. The goal of Samuel, however, was to produce good results, while the goal of Newell, Simon, and Shaw was to simulate human problem-solving techniques as well as to produce acceptable results. In addition, game playing is only one area of Artificial Intelligence. The success of category two, the any method possible approach, in comparison to the human simulation approach in game playing is not sufficient evidence upon which to decide which approach to Artificial Intelligence is better. Different approaches may have different comparative results when applied to the whole spectrum of Artificial Intelligence. Research in both categories is continuing as well as research in combinations of the two, and sufficient information should be gathered to permit better conclusions to be made in the future.

II. THE GAME PLAYING PROGRAM

A. OBJECTIVE

The primary objective of this project is to develop a game playing program. The program must generate moves that would be considered intelligent if made by a human being. The approach used in the game playing program favors the philosophy of Newell, Simon, and Shaw. It does not, however, belong distinctly to either of the categories. It is, instead, a combination of the two. As in Samuel's work, the criterion of success in this project will be the quality of the results obtained. The basic program structure, however, will be patterned after the Newell, Simon, and Shaw design. The program will attempt to duplicate the overall strategic approach used by humans. The program is intended to generate moves based on the same considerations human players appear to use.

The game chosen to provide the environment for research is the game of GO. GO was chosen because it cannot be handled by the brute-force method. In addition, less research has been done on GO than on CHESS or CHECKERS. Because little research has been done in GO, there remains a greater spectrum of game playing techniques to be explored. Many techniques have already been proven feasible or unfeasible in relation to CHESS or CHECKERS. These same techniques may produce different results in the GO environment.

B. BACKGROUND

The game of GO, known as IGO in Japanese and WEI-KI in Chinese, is the oldest game of strategy in the world. It has survived the trial of centuries without the rules being changed in any significant respect. Three times as old as the modern version of CHESS, the game had its beginning around 2000 B.C. in China. Today the game of GO is the national game of Japan and, along with CHESS, is a principal game of strategy in the occidental world.

The game in general is patterned after the ancient principals of war, the objective of which is to occupy and defind territory while simultaneously capturing the enemy and his territory. The battlefield consists of a game board containing 19 vertical and 19 horizontal lines drawn parallel to the edges. Unlike CHESS or CHECKERS, the moves are made on the intersections of the lines rather than the squares formed by the lines. There are 361 intersections in all. A move consists of placing a white or black stone on the board. Each player has 181 stones but seldom are all of them used in any particular game. Once a stone is placed on the board, it cannot be moved.

The primary objective of the game is to join stones gradually into groups surrounding as many vacant intersections as possible. The secondary objective is to capture as many hostile stones as possible. This is done by occupying all the vacant points directly adjoining an opponent's stone or group of stones. If the opponent's stones are completely surrounded, the stones are removed from the game board and are classified as prisoners.

The game is over when neither player can make a move that increases the size of his territory or the number of his prisoners. Each player then scores the number of vacant points he has surrounded, less the number of his men captured by his opponent. The higher score wins the game.

C. STRATEGY

In competitive games or sports, human beings tend to favor a particular strategy. Like politics, where a person cannot only be designated a Republican or Democrat but a conservative or liberal as well, game strategy is best categorized as a two-dimensional plane.

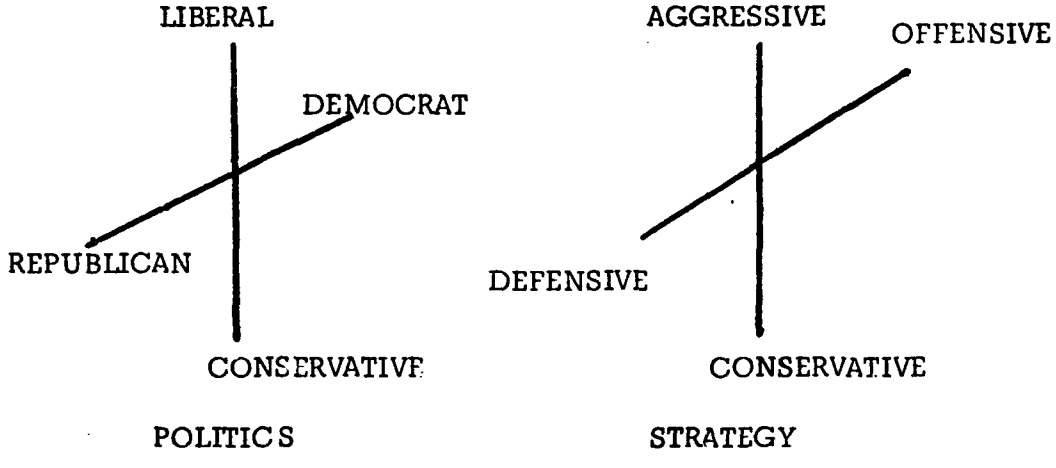

Figure 1.

The first division of strategies is concerned with the degree of aggressiveness. An aggressive player generally initiates all the action and sets the pace of the game. His play entails a high degree of risk but can produce beneficial results when properly executed. The

counterpart of the aggressive player is the conservative. The conservative player places his emphasis on consolidating a position and slowly exploring new areas of play. The conservative player allows the action to be initiated by his opponent. The degree of risk taken by a conservative player is low. His success, however, depends on the opponent making an error.

The strategies can also be divided into offensive and defensive categories. The offensive strategies emphasize the scoring of points while the defensive strategies emphasize the prevention of scoring. The offensive player bases his moves on his own previous moves and preplanned objective; he does not consider the moves of his opponent. The defensive player, like the conservative, allows the action to be initiated by his opponent. For each move or play made by his opponent, the defensive player counters with a move of his own. The objective of the defensive player's move is to minimize the effect of his opponent's move on the game situation.

The actual strategy used by any particular game player lies in one of the four sectors of the two-dimensional plane. The actual strategies can be classified as offensively aggressive, defensively aggressive, offensively conservative, or defensively conservative. The overall strategy of this project's game player belongs to the defensively aggressive category. The program is classified as such for the following reasons:

1. The program allows the opponent to initiate the action.

2. The primary objective of the program is to reduce the scoring potential of the opponent.

3. The program attempts to prevent the joining of the opponent's stones into dangerous patterns.

4. Unlike the conservative player, the program does not consolidate its own position. Wherever a threat exists he attacks.

5. Analysis prior to generating a move always tests for the most dangerous situations first.

6. The investigation of the board initially begins at the location of the opponent's last move to determine if a counter move is required.

Seldom do humans exercise an extreme strategy. Their actual strategy is near the origin of the plane, with the emphasis placing the strategy in one of the four categories. Similarly, the game playing program does contain some offensive procedures. The emphasis, however, is on the premise that the best offense is a good defense.

The defensively aggressive strategy is implemented by a set priority system. The priority system cannot be altered during the play of the game. The program itself can play either Black or White. The actual priority system is embodied in the procedure ANALYZE, which always considers the greatest threats first.

D. PATTERN-MATCHING

Pattern-matching, as it applies to this program, is the ability to recognize distinct patterns of stones on the game board. The game of GO has many distinct patterns of stones which can be classified as either advantageous or dangerous. In playing the game, a human being can often recognize some of these patterns and respond accordingly. This recognition reduces the degree and depth of analysis prior to a move. Whenever an obvious pattern exists, the person stops analyzing the situation and makes a move. Therefore, it was advantageous to include a pattern-matching scheme in the program to provide a mechanism for selecting moves. Without this mechanism, it is necessary to propose various moves, and consider the relative advantages gained in the resulting situations. Pattern-matching results in a decrease in response time, as well as simulating the human use of patterns in selecting moves.

The general pattern-matching technique contained in the program is similar to the technique used by Uhr in playing TIC-TAC-TOE. That is, a list of dangerous or advantageous patterns is stored in the memory of the machine. In the present program, these patterns are stored in COMPARE and CHECK. During the execution of the procedures SCAN1 through SCAN8, two patterns are created. They are constructed in the form of one-dimensional arrays. One array is called PATT and consists of the locations of the opponent's stones in a certain 45 degree sector. The other array is known as OFF and contains the locations of the machine's stones in the same 45 degree sector.

At the completion of each scan routine, the actual patterns are compared to those stored in memory. In keeping with the basic defensive strategy of the program, the pattern, PATT, is compared first. If either PATT or OFF matches a stored pattern, appropriate action is taken by calling BLOCK or BUILD, respectively.

Because of the number of dangerous or advantageous patterns which might arise in GO, it is beneficial to program the machine to generalize patterns. The machine should be programmed to some extent to recognize the effect of rotation and reflection on specific patterns. This will greatly reduce the number of patterns that must be compared and hence the search and response time will be similarly reduced. A detailed explanation of the generalization technique used by the game playing program appears in the procedure SCAN.

E. GENERATING A MOVE

The name of the game playing program is GO. It is the responsibility of this procedure to interact with an opponent and play a game. The initial move of the game can be done by either the machine or its opponent. Subsequently, the machine can play either black or white. The internal representation is concerned with the machine's move and the opponent's move, regardless of color. If the machine is to make the first move, the procedure GO calls OPENING, which makes a standard book move. Following the initial move, all subsequent moves by the machine are generated by the procedure ANALYZE.

The moves generated by ANALYZE fall into two categories. The first category contains moves which are generated by special cases. These include responses to known patterns of stones, situations of extreme danger, and standard opening moves that are taught to beginning students of the game. The second category contains the majority of the moves generated during a game. These moves are generated by using a set of parameters and require a greater response time due to the increased depth of analysis. Most moves which occur in the second category are generated by COUNTER. The procedure COUNTER is responsible for making the move which will prevent the opponent from strengthening his board position.

The procedure ANALYZE is called by GO with data giving the position of the machine's and the opponent's last moves. ANALYZE first investigates the possibility of immediate danger to one or more of the machine's stones. Danger exists when the opponent can capture one or more of the machine's stones on his next move. If danger exists, the control of the move-generator is shifted to the procedure ESCAPE. ESCAPE determines whether a move can be made which will prevent the capture of the stones. The list of all possible escape moves is then explored and the best move is made. If no move exists, control is returned to ANALYZE.

If the game is in its early states -- that is, less than 18 moves have expired -- control is given to the procedure OPENING. This procedure performs some of the basic opening moves taught to beginning

students of the game. These moves are built into the program; the computer is not required to learn these moves by trial and error. If OPENING can make a beneficial move, the move is made and control of the game is returned to GO. If no move is made, control is assumed by ANALYZE.

The next step performed by ANALYZE is to gather information and data to be used for analysis. This is done by calling SCAN. SCAN is given the board intersection upon which the analysis is to be done. This board intersection is referred to as the vertex of the scan procedures. The function of SCAN is to conduct an investigation on the board area adjacent to the vertex of the scan routines.

The first board intersection given to SCAN corresponds to the opponent's last move. The object of this scan is to determine the reason for the move just made by the opponent. Since moves in GO usually occur in series, if the last move can be understood then the opponent's next move can be anticipated.

Humans often use this same approach when playing games of strategy. The player studies the last move made by his opponent and attempts to understand the opponent's logic. If he can understand why a particular move was made, he can anticipate the next move of his opponent. He can then generate a move which will make the opponent's next logical move ineffective. Both approaches emphasize the understanding of the opponent's move. This correlation between the machine's

approach to the priority of board analysis and a human approach indicates that the game playing program is in the category of human simulation.

The actual investigation is performed by the procedures SCAN1 through SCAN8, which are called by SCAN. These procedures are all basically alike, with each investigating a different 45 degree sector of the area adjacent to the vertex of the scan. Each procedure looks at 20 intersections. The combination of these eight procedures gives a composite picture of the situation in the area of the board adjacent to the vertex. The composite picture contains information from 121 board intersections. These board intersections form a square, 11 intersections by 11 intersections.

As in most board games, there exist in GO certain patterns which are either advantageous or dangerous. If during the execution of SCAN one of these distinct patterns is found, the investigation is terminated and a response to the pattern is generated. If the pattern found is dangerous to the program, control of the move-generator is given to the procedure BLOCK. If a move to counter the pattern is available, it is generated. If the pattern is advantageous to the machine, control is given to BUILD which attempts to reduce the effect of the opponent's last move on the pattern. This can be done either by consolidating the machine's position or by isolating the opponent's stone and rendering it ineffective. If the opponent's last move did not effect the advantageous pattern, control is returned to SCAN.

If the investigation produces no specific patterns, and the composite picture produced by SCAN is completed, the next step in ANALYZE is determined by a set of parameters. The values of these parameters are obtained from data collected by SCAN. Using these parameters, a choice is made as to whether the action of the game should or should not be transferred to a different area of the game board. This transfer would take place if control of the current board area is firmly in the hands of one player or the other. The degree of control is determined by the value of the parameter VAL.

If the action of the game is not transferred to another area of the board, a response to the opponent's move is generated by COUNTER. COUNTER will always make a move.

If the opponent's last move presents no threat and the control of this area is firmly in the hands of one player or the other, SCAN is called with the location of the machine's last move. Should it again not be advantageous to make a move in this area of the board, control is given to INITIATE. The procedure INITIATE searches new areas of the board until one is found where an advantageous move can be made. When this area is found, a move is generated by COUNTER and the control of the program is returned to GO to wait for the opponent's next move.

F. THE GAME PLAYING PROCEDURES

The procedures used in this game playing program can be divided into three separate categories; fundamentals, analysis, and move-generation.

The fundamental category contains all the procedures which perform basic functions needed to play a game. These are MOVE, COUNTERMOVE, PATTERN, INITIAL, and UPDATE. Fundamental procedures are called by other procedures; they do not directly affect the playing of the game. The analysis category contains all the procedures which perform analysis and control the flow of the game. These procedures gather data and make various decisions during the play of the game. They do not generate specific moves but do determine what procedure will have the responsibility for the final move generation. Members of this category are ANALYZE, SCAN, COUNTER, SCAN1 through SCAN8, BLOCK, BUILD, COMPARE, CHECK, and INITIATE. The third category, move-generation, contains all the procedures which generate specific moves. Move-generation procedures do perform some analysis but are primarily responsible for the specific move generated by the program. This category consists of LOOK1 through LOOK4, COUNTER1 through COUNTER8, OPENING, and ESCAPE.

1. Procedure INITIAL

The procedure INITIAL performs the initial setup for the game. It places the value zero, indicating the empty state, in all the board positions and initializes all the logical and numerical parameters.

2. Procedures MOVE and COUNTERMOVE

The function of MOVE and COUNTERMOVE is to update the game board to reflect the making of a move. The procedure MOVE updates the board after a move is made by the machine. This is done by

placing a 1 in the memory location corresponding to the board position affected by the machine. The procedure COUNTERMOVE updates the board representation after a move is made by the machine's opponent. The procedure COUNTERMOVE places a 2 in memory vice a 1. MOVE, COUNTERMOVE, and UPDATE are the only procedures which can directly change the status of the game board.

3. Procedure UPDATE

The function of the procedure UPDATE is to change the board representation to reflect the capturing of prisoners. UPDATE places a zero in the memory location representing the board position or positions affected by the capture. The argument of UPDATE is a variable representing the number of stones captured. After the execution of COUNTERMOVE, the machine prints the word UPDATE. If a stone or stones have been captured, the opponent types the number of captured stones. The program will ask for the xy-coordinates of each. If the number typed is zero, UPDATE is bypassed.

4. Procedure OPENING

The function of OPENING is to initiate action at the beginning of the game. OPENING contains a list of opening moves usually taught to beginning students of the game. The procedure can only be entered if less than 18 moves have been made. The entering of OPENING is controlled by the parameter COUNT. COUNT keeps count of the number of moves made by the machine. The parameter COUNT is augmented by one each time the procedure ANALYZE is called.

5. Procedure INITIATE

The procedure INITIATE has as its main function the expansion of the game into less active areas of the game board. Given a certain board position, INITIATE will evaluate the adjacent area with the aid of the scan routines. Using the parameter VAL, obtained from the overall strength or weakness of the investigated area, INITIATE will decide if a beneficial move can be made in this area or if another area of the board should be investigated.

6. Procedure ESCAPE

The function of the procedure ESCAPE is to remove the machine's stones from the immediate danger of being captured. ESCAPE is initiated by the procedure ANALYZE. In the game of GO, as with the KING in CHESS, if a stone is in immediate danger of being captured, that stone's player must be informed. The machine is informed of such danger by the logical variable DANGER. The variable DANGER is set to true or false by the machine's opponent. When the variable DANGER is true, ANALYZE transfers control to the procedure ESCAPE. ESCAPE then searches the area adjacent to the endangered stone and looks for an opening by which the endangered stone can be removed from immediate danger. If more than one escape route exists, ESCAPE determines the best. This would correspond to removing the KING from "check" in the game of CHESS. If no escape can be found, the control of the program is returned to ANALYZE.

7. Procedure BLOCK

The primary function of the procedure BLOCK is to counter the opponent's efforts to surround and control vacant intersections. BLOCK is called by two separate procedures. If a dangerous combination of the opponent's stones is found during pattern-recognition, control is sent to BLOCK from the scan routines. The procedure BLOCK then investigates the area adjacent to the two end stones of the pattern. The counter move is made at the area adjacent to an end stone where the machine's stone can be most easily defended from capture. In order for a blocking tactic to be effective, it must originate in an area of strength, not weakness.

In addition to blocking dangerous patterns, BLOCK is also called from the procedure COUNTER. If in COUNTER it is determined that two of the opponent's stones are in a position to link with other opponent's stones, BLOCK is called to prevent this linkage from occurring. The locations of the two stones are sent to BLOCK and the investigation is conducted on these two locations as it is with the two end positions of a pattern. The countermove is made with the same considerations as before. By using this approach, the machine is often able to stop the formation of dangerous patterns while simultaneously strengthening its own position.

8. Procedures COMPARE and CHECK

The procedures COMPARE and CHECK are directly related to each other. The function of both is to check for specific patterns found

by the procedure SCAN. COMPARE tests for dangerous patterns, while CHECK tests for advantageous patterns. The only significant difference between the two procedures is that CHECK is entered with the array OFF while COMPARE is entered with the array PATT. The division of pattern-recognition into two procedures was done to improve response time by reducing the number of patterns to be compared. Patterns of the machine's stones are compared only to the list of advantageous patterns located in CHECK. Patterns of the opponent's stones are compared to the list of dangerous patterns located in COMPARE.

The internal structure of COMPARE and CHECK are the same. Each contains a list of important patterns stored in the machine's memory. Patterns can be added or deleted from these lists without significant difficulty. The additions and deletions, however, must be done manually. The game playing program in its present state does not have the capability to add or delete on its own.

9. Procedure BUILD

The procedure BUILD performs the opposite function to the procedure BLOCK. The main objective of BUILD is to join the machine's stones into chains which will surround and control vacant intersections. The procedure is called from two separate segments of the game playing program. The most frequent reference to BLOCK is made by COUNTER. The purpose in calling BUILD from COUNTER is to start new offensive chains. COUNTER sends to BUILD the location of two positions which are to be formed into a chain. If the two locations are at a distance

greater than two, a stone is placed between them to make a more secure position. If the distance is less than two, an investigation is conducted at the two locations to determine how best to extend the chain. Consideration is given to the strength and weakness of the adjacent areas and the closeness of the edges of the game board. In GO the edges of the board are considered natural fortifications. These natural fortifications can be used as part of the chains which surround vacant intersections. By using the edges as part of the chain, the number of stones needed to surround a vacant area is greatly reduced.

The other segment of the game playing program which calls BUILD is the procedure CHECK. If CHECK finds an advantageous pattern, it calls BUILD to extend this pattern for greater strength. This is done by giving to BUILD the two end positions of the pattern. An investigation is conducted at these locations as it is when BUILD is called by COUNTER. The investigation is to determine the best way to extend the pattern, again giving consideration to the relative strength and weakness of the area and the closeness of the edges of the game board.

10. Procedure PATTERN

PATTERN obtains representation of the machine's stones and the opponent's stones in one 45 degree sector for use by the procedure SCAN. PATTERN is referenced by the procedures SCAN1 through SCAN8. The procedure PATTERN constructs two one-dimensional arrays called PATT and OFF. PATT is a one-dimensional array that shows the location

of the opponent's stones relative to the searching technique used by the scan routines. OFF is a one-dimensional array that shows the location of the machine's stones relative to the searching technique used by the scan routines.

In the search of a 45 degree sector, the order of progression is always from the horizontal or vertical axis to the 45 degree diagonal. Each increment on the horizontal or vertical axis is considered a distance of one. The scan begins at the vertex and looks at all intersections of a certain distance before incrementing the distance by one.

If the intersection currently being looked at contains an opponent's stone, the corresponding element of PATT is set to the value of two while the corresponding element of OFF is set to zero. If the intersection contains a machine's stone, the corresponding element of PATT is set to the value 0 while the corresponding element of OFF is set to 1. If the intersection is empty, the corresponding elements of both PATT and OFF are set to the value 0.

11. Procedure COUNTER

The procedure COUNTER is the primary move generator of the game playing program. When none of the special cases or pattern-recognition schemes apply, the responsibility for making a move is given to COUNTER. The function of COUNTER is to consider all good moves that will have an effect on the last move of the opponent and decide what move is best.

COUNTER receives data giving the following information from SCAN:

1. The position of the vertex of the scan routines.
2. The location of the machine's stone closest to the vertex and the distance from the vertex to this stone.
3. The location of the opponent's stone closest to the vertex and the distance from the vertex to that stone.

Using this data, COUNTER determines what move should be made in response to the opponent's last move.

If the vertex of the scan routines is vacant, COUNTER generates a move at that location. This condition could arise only if COUNTER was called from INITIATE. The board intersections used by INITIATE are all strategically located and occupation of these locations is always beneficial to the machine.

If the vertex is not vacant, COUNTER conducts an analysis to determine the positions of the machine's and the opponent's closest stones relative to the vertex. Each of the stones is classified according to the position it maintains in relation to the vertex (see Figure 2). The board area adjacent to the vertex is divided into eight sectors numbered consecutively beginning in the lower right sector. The relative location is used in conjunction with the previous information obtained from SCAN to determine the next step to be taken by COUNTER.

If the vertex contains a machine's stone, the responsibility for generating the next move is given to BUILD. BUILD is called with

data giving the location of the vertex, the location of the machine's stone closest to the vertex, and the sector number representing the position of the machine's closest stone relative to the vertex.

If the intersection contains an opponent's stone, COUNTER chooses between two alternatives. If the distance between the vertex and the opponent's closest stone is less than three, the responsibility for generating a move is given to BLOCK. BLOCK is called with data giving the location of the vertex, the actual location of the opponent's closest stone, and the number representing the position of the opponent's closest stone relative to the vertex. If the distance is three or greater, the move is generated by one of the counter routines, COUNTER1 through COUNTER8. The COUNTER routine called is determined by the number representing the relative location of the opponent's closest stone to the position of the vertex.

12. Procedure SCAN

The function of SCAN is to conduct an investigation around a given board intersection. SCAN is classified as an analysis procedure and controls the calling of the procedures SCAN1 through SCAN8. Each of the procedures, SCAN1 through SCAN8, analyzes a different 45 degree sector of the game board out to a distance of five horizontal or vertical lines. The maximum number of intersections looked at by each scan routine is twenty. The scan is conducted from a given board intersection, known as the vertex, to all intersections at the distance of five from the vertex. The distance of five was chosen because in

most instances the influence of a stone does not extend past that range.

In searching a particular sector, the order of progression is always from the horizontal or vertical axis to the 45 dégree diagonal. By combining the eight scanned sectors into one composite picture, an area of the board eleven horizontal by eleven vertical lines is constructed with the center of the area located at the vertex of the scan routines. The composite picture contains information from 121 intersections. The information obtained from the scan procedures is used by ANALYZE and COUNTER to decide what response to the opponent's last move should be made.

During the scan of each 45 degree sector, SCAN1 through SCAN8 call the procedure PATTERN, which constructs the two one-dimensional arrays known as PATT and OFF. These are constructed by placing a 2,1, or zero in the corresponding element of the array which represents the intersection currently being looked at. At the conclusion of each scan of a 45 degree sector, the arrays PATT and OFF are sent to COMPARE and CHECK, respectively, to test for advantageous or dangerous patterns. If an advantageous or dangerous pattern is found, SCAN informs ANALYZE, which calls BUILD or BLOCK to generate a response.

The actual construction and testing of patterns requires further explanation. The elements of the arrays PATT and OFF are numbered 1 to 14. They correspond to the first through fourteenth

intersections looked at during the scan of each sector (see Figure 3). Since the scan is conducted similarly in all of the 45 degree sectors, the representation of a particular pattern located in one sector resembles a similar pattern located in another sector. Two patterns are considered similar if one is the reflection of the other or if one can be obtained from the other by rotating the pattern around the vertex of the scan routines (see Figure 4).

Consider all the representations of the pattern in figure 4a. The pattern consists of four stones with three in a row. By taking all the reflections and rotations of this pattern, we can produce 8 representations of the pattern in sectors one and two of the procedure SCAN. Since SCAN has 8 sectors, the total number of representations of the pattern would be 32. Because the scan is the same in each sector, the 32 representations can be obtained by rotating one of the eight representations in sectors one and two about the vertex. It is, therefore, only necessary to store eight representations of the pattern in the machine's memory. This reduces the stored patterns by a factor of four.

Another illustration of the pattern representation would be that eight linear formations of the same number of stones can be found in these eight sectors. These patterns are either parallel to the edges of the game board or at 45 degree angles to the edges. The pattern-matching technique of the game playing program considers all the patterns parallel to the game board edges to be the same. Likewise, it considers all of the patterns at 45 degree angles to be the same. The

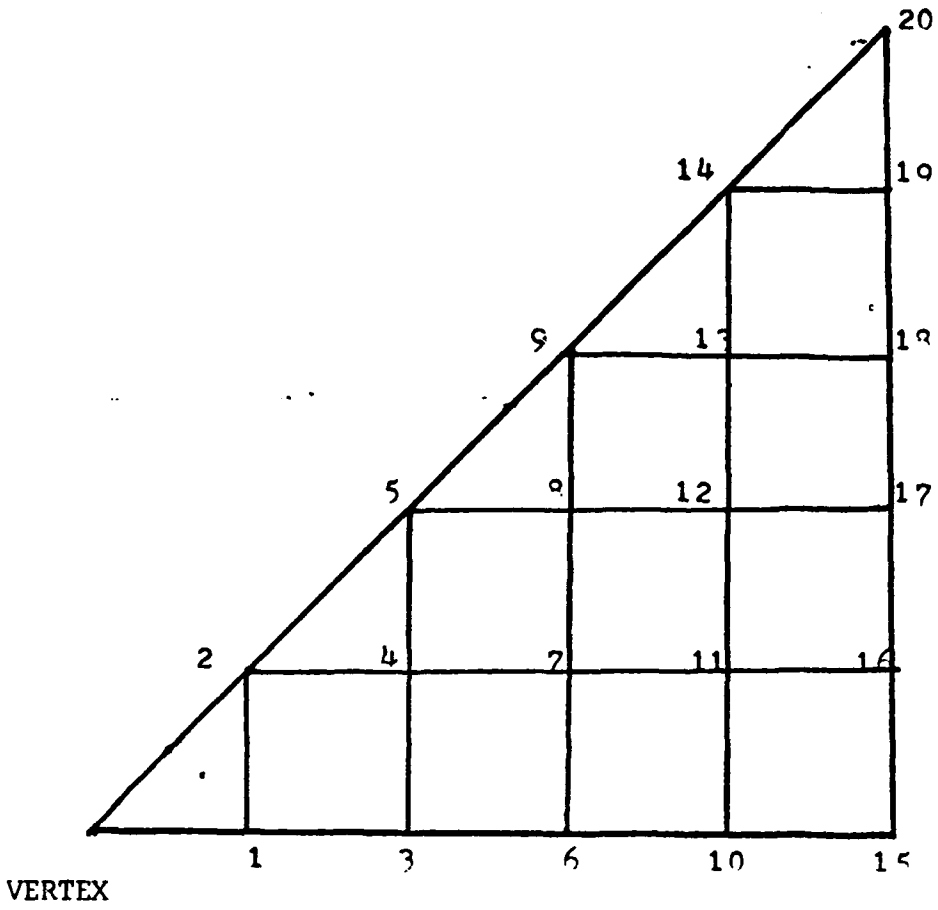

Figure 3.

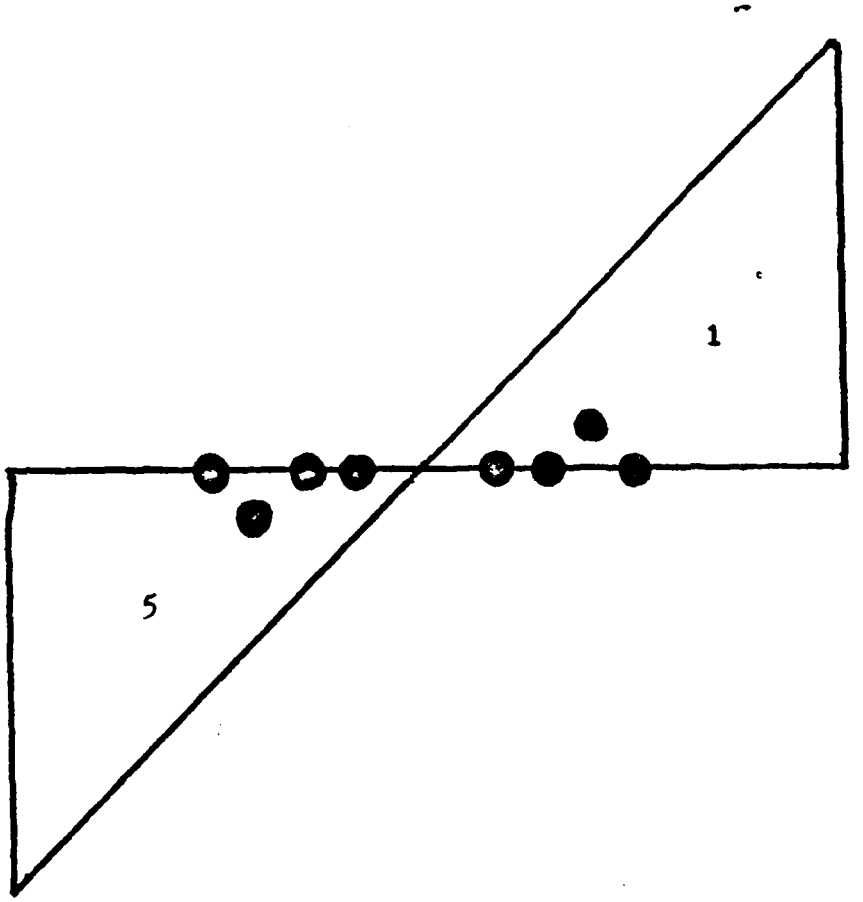

Figure 4.

eight actual patterns are now represented in memory by two. This reduces the number of linear patterns that must be compared by four. Consider the pattern in Figure 5. If this particular pattern was constructed by SCAN1, it would represent four of the opponent's stones in a linear formation parallel to the bottom edge of the game goard (see Figure 6). If this same pattern was constructed by SCAN3, it would represent four of the opponent's stones in a linear formation perpendicular to the bottom edge of the game board (see Figure 7). Since the important aspect of the pattern is the linear formation, not whether it is parallel or perpendicular, it is in essence the same pattern and should be identified and handled in the same manner.

The advantages to this type of pattern-matching are that it is relatively easy to implement, gives good response time, and significantly reduces the number of patterns that must be represented in the machine's memory. A future improvement to the program would be to devise a more efficient scheme that could reduce the quantity of stored patterns even more. The disadvantage of the present scheme is that reflections and rotations within an individual scan sector are not recognized by the program.

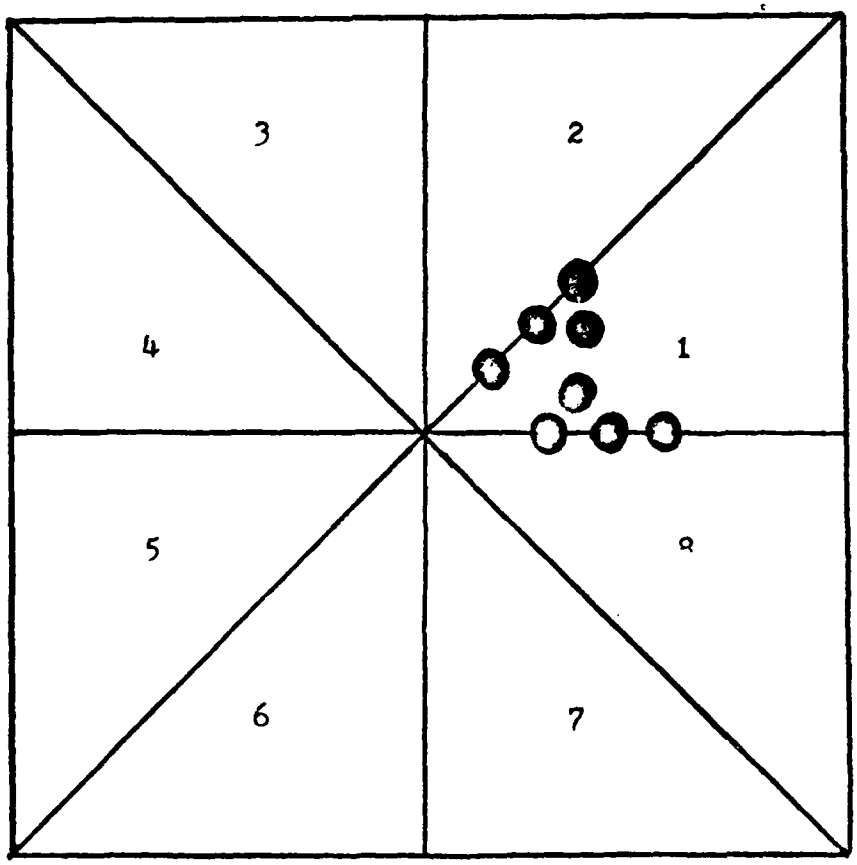

Figure 4a.

(2,0,2,0,0,2,0,0,0,2,0,0,0,0)

Figure 5.

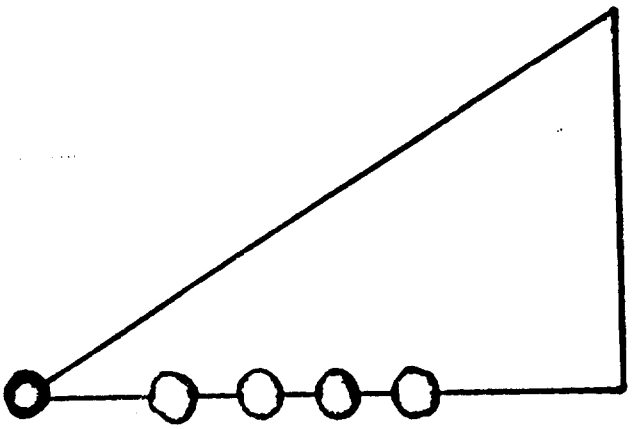

Figure 6.

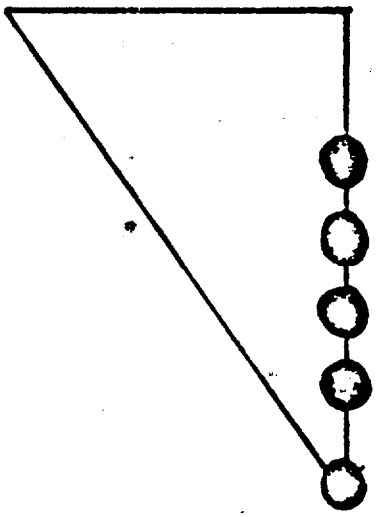

Figure 7.

III. OBSERVATIONS AND CONCLUSIONS

The objective of this project was to develop a game playing program that played the game of GO. This objective was obtained. The program plays the game and generates moves using analysis and generalization. The program has obtained a level of play that would be considered reasonable by a human player. The program did not, however, reach a level much advanced over a beginning player.

The defensively aggressive strategy used by the program accomplished some interesting results. The scores of the games played with the program were lower than those considered average game scores. The average game score is between 90 and 120. The scores of the games played with the program averaged between 70 and 80. Since the objective of the defensive strategy was to reduce the number of points scored, it can be concluded that the program's defensive strategy met with reasonable success.

One of the scoring methods in GO is to capture prisoners. Unfortunately, the number of machine's stones captured was fairly high while the number of the opponent's stones captured was low. This can be attributed to the aggressiveness of the program. The program, in order to block its opponent from linking his stones into chains, often generated a move that had a high degree of risk, leaving it vulnerable. An improvement in the strategy might be to reduce the

aggressiveness. This could be accomplished by adjusting the parameters STRENGTH, WEAKNESS, and VALUE.

Another weakness in the strategic approach is that the program concentrates on blocking the moves of its opponent, rather than surrounding the opponent's stones. The defensive strategy placed the emphasis on the prevention of the opponent surrounding vacant intersections rather than capturing prisoners. An improvement in the program would be to add a procedure that would concentrate on surrounding the opponent's stones. This would increase the number of the opponent's stones taken prisoner. This improvement would not contradict the defensive strategy as long as the priority of the new procedure was low.

The set priority system implemented by ANALYZE was designed to provide a defensive strategy. Samuel discovered in his CHECKER-playing program that the priority attached to the goals changed during the play of the game. In order to play a successful game of CHECKERS, it was necessary to provide a mechanism for altering the priority of goals during the game. The ability to alter the priority was not necessary in GO. The set priority system did provide a defensive strategy that was reasonably successful as previously stated. The efficiency of the priority system was the same at the beginning of the game as it was in the middle and at the end. The only modification to the defensive strategy was to include the procedure OPENING that made standard book moves. This is an example where the conclusions made in one game environment, altering priorities, does not have the same effect or value in another game environment.

The original design plan for the game playing program did not call for any learning schemes to be implemented. In the development of a pattern-matching capability for the program, however, the technique used set the foundation for a learning capability. This foundation is based on the ability to manually add or delete patterns to the procedures COMPARE and CHECK. This corresponds to the human's ability of being taught the advantage or disadvantage of a pattern by another person. Although the patterns must presently be added or deleted manually, it should not be too great a task to allow the program to discover and add patterns on its own. Whether the patterns are added manually or by the program, both represent a type of learning according to Minsky (1968, p. 14).

LIST OF REFERENCES

Fiegenbaum, Edward A., and Julian Feldman, Eds. Computers and Thought. New York, McGraw Hill Book Company, 1963.

Lasker, Edward. GO and GO-MOKU. New York, Dover Publications, Inc., 1960.

Minsky, Marvin, Ed. Semantic Information Processing. Cambridge, Massachusetts: The MIT Press, 1968.

Newman, C., Uhr, L., "BOGART-A Discovery and Induction Program for Games." ACM Twentieth National Conference, 1965.